

# Package ‘delma’

May 14, 2025

**Type** Package

**Title** Convert 'R Markdown' and 'Quarto' Documents to Ecological Metadata Language

**Version** 0.1.1

**Description** Ecological Metadata Language or 'EML' is a long-established format for describing ecological datasets to facilitate sharing and re-use. Because 'EML' is effectively a modified 'xml' schema, however, it is challenging to write and manipulate for non-expert users. 'delma' supports users to write metadata statements in 'R Markdown' or 'Quarto markdown' format, and parse them to 'EML' and (optionally) back again.

**Depends** R (>= 4.3.0)

**Imports** cli, dplyr, glue, lightparser, purrr, quarto, rlang,  
rmarkdown, snakecase, stringr, tibble, withr, xfun, xml2

**Suggests** knitr, testthat (>= 3.0.0)

**License** GPL-3

**URL** <https://delma.ala.org.au>

**BugReports** <https://github.com/AtlasOfLivingAustralia/delma/issues>

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Martin Westgate [aut, cre],  
Shandiya Balasubramaniam [aut],  
Dax Kellie [aut]

**Maintainer** Martin Westgate <[martin.westgate@csiro.au](mailto:martin.westgate@csiro.au)>

**Repository** CRAN

**Date/Publication** 2025-05-14 09:00:01 UTC

Contents

as_eml_list . . . . .	2
as_eml_tibble . . . . .	3
as_eml_xml . . . . .	4
as_lp_tibble . . . . .	5
check_metadata . . . . .	6
read_eml . . . . .	7
read_md . . . . .	7
use_metadata_template . . . . .	8
write_eml . . . . .	9
write_md . . . . .	10
<b>Index</b>	<b>11</b>

---

as_eml_list	<i>Convert metadata to a list</i>
-------------	-----------------------------------

---

Description

Takes an object of class `xml_document` or `tibble`, and converts it to a list. When converting from an `xml_document`, this is simply a wrapper for `xml2::as_list()`

Usage

```
as_eml_list(x, ...)  
  
## S3 method for class 'tbl_lp'  
as_eml_list(x, ...)  
  
## S3 method for class 'tbl_df'  
as_eml_list(x, ...)  
  
## S3 method for class 'list'  
as_eml_list(x, ...)  
  
## S3 method for class 'xml_document'  
as_eml_list(x, ...)
```

Arguments

- `x`                      Object to be converted
- `...`                   Other arguments, currently ignored

Value

A list, where both the nested structure of the XML/md and the attributes of XML nodes, are preserved.

**Examples**

```
source_file <- system.file("extdata",
                           "bionet_metadata.Rmd",
                           package = "delma")
df <- read_md(source_file)
as_eml_list(df) |> str()
```

---

as_eml_tibble	<i>Convert metadata to a tibble</i>
---------------	-------------------------------------

---

**Description**

Takes objects of class `list` or `xml_document` and converts them to a tibble with a particular structure, designed for storing nested data. Tibbles are required because attributes are stored as list-columns, which are not supported by class `data.frame`.

**Usage**

```
as_eml_tibble(x, ...)

## S3 method for class 'tbl_df'
as_eml_tibble(x, ...)

## S3 method for class 'tbl_lp'
as_eml_tibble(x, ...)

## S3 method for class 'list'
as_eml_tibble(x, ...)

## S3 method for class 'xml_document'
as_eml_tibble(x, ...)
```

**Arguments**

<code>x</code>	Object to be converted
<code>...</code>	Other arguments, currently ignored

**Value**

An object of class `tbl_df`, `tbl` and `data.frame`, containing the following fields:

- `level` (int) gives the nestedness level of the node/heading in question
- `label` (chr) the xml tag
- `text` (chr) Any text stored within that tag
- `attributes` (list) Any attributes for that tag

**Examples**

```
source_file <- system.file("extdata",
                           "bionet_metadata.xml",
                           package = "delma")
xml_data <- xml2::read_xml(source_file)
as_eml_tibble(xml_data)
```

---

as_eml_xml	<i>Convert metadata to an xml_document</i>
------------	--

---

**Description**

Takes a character vector, tibble, or list and converts it to an `xml_document`, as defined by the `xml2` package. When converting from a list, this is simply a wrapper for `xml2::as_xml_document()`.

**Usage**

```
as_eml_xml(x, ...)

## S3 method for class 'tbl_lp'
as_eml_xml(x, ...)

## S3 method for class 'tbl_df'
as_eml_xml(x, ...)

## S3 method for class 'list'
as_eml_xml(x, ...)

## S3 method for class 'xml_document'
as_eml_xml(x, ...)
```

**Arguments**

<code>x</code>	Object to be converted.
<code>...</code>	Other arguments, currently ignored.

**Value**

An `xml_document` with the specified nodes and attributes.

**Examples**

```
source_file <- system.file("extdata",
                           "bionet_metadata.Rmd",
                           package = "delma")
df <- read_md(source_file)
as_eml_list(df) |> str()
```

---

as\_lp\_tibble*Convert metadata to a `lightparser` tibble*

---

### Description

Takes objects of class `tbl_df`, `list` or `xml_document` and converts them to a tibble with a structure required by `lightparser`. Note that `delma` represents these as an object of class `tbl_lp` for convenience.

### Usage

```
as_lp_tibble(x, ...)  
  
## S3 method for class 'tbl_lp'  
as_lp_tibble(x, ...)  
  
## S3 method for class 'tbl_df'  
as_lp_tibble(x, ...)  
  
## S3 method for class 'list'  
as_lp_tibble(x, ...)  
  
## S3 method for class 'xml_document'  
as_lp_tibble(x, ...)
```

### Arguments

<code>x</code>	Object to be converted.
<code>...</code>	Other arguments, currently ignored.

### Value

An object of class `tbl_lp`, `tbl_df`, `tbl` and `data.frame`, containing the following fields:

- `type` (chr) Whether that section is e.g. YAML, inline text, heading, or code block
- `label` (chr) The tag associated with a given code block (otherwise NA)
- `params` (list) Attributes of a code block
- `text` (list) Any text in that section
- `code` (list) Any code in that section
- `heading` (chr) For `type = heading`, the value of that heading
- `heading_level` (dbl) The heading level of that heading (i.e. number of #)
- `section` (chr) The heading this section sits within

Examples

```
source_file <- system.file("extdata",
                           "bionet_metadata.xml",
                           package = "delma")
xml_data <- xml2::read_xml(source_file)
as_lp_tibble(xml_data)
```

---

check_metadata	<i>Check validity of a metadata statement</i>
----------------	---

---

Description

In the Darwin Core standard, metadata statements are mandatory and must be provided in Ecological Metadata Language (EML) in a file called eml.xml. This function applies a series of checks designed by GBIF to check the structure of the specified xml document for consistency with the standard. Note, however, that this function doesn't check the *content* of those files, meaning a file could be structurally sound and still be lacking critical information.

Usage

```
check_metadata(file = NULL, schema = NULL, quiet = FALSE)
```

Arguments

file	An EML file to check Can be either local or a URL.
schema	Either NULL (the default) to compare to the GBIF profile; or a URL to a valid schema (passed internally to <a href="#">xml2::read_xml</a> ).
quiet	(logical) Should messages be hidden? Defaults to FALSE.

Details

This function uses local versions of dc.xsd, eml-gbif-profile.xsd and eml.xsd downloaded from <http://rs.gbif.org/schema/eml-gbif-profile/1.3/> on 2024-09-25.

Value

Invisibly returns a tibble showing parsed errors; or an empty tibble if no errors are identified.

Examples

```
source_file <- system.file("extdata",
                           "bionet_metadata.xml",
                           package = "delma")
check_metadata(source_file)
```

---

read_eml	<i>Read an EML-formatted metadata document</i>
----------	--

---

**Description**

read\_eml() imports metadata from an EML file into the workspace as a tibble.

**Usage**

```
read_eml(file)
```

**Arguments**

file	Filename or URL to read from.
------	-------------------------------

**Value**

read\_eml() returns an object of class tbl\_df, tbl and data.frame (i.e. a tibble).

**Examples**

```
source_file <- system.file("extdata",  
                           "bionet_metadata.xml",  
                           package = "delma")  
df <- read_eml(source_file)
```

---

read_md	<i>Read markdown-formatted metadata</i>
---------	---

---

**Description**

read\_md() imports metadata from a markdown file into the workspace as a tibble.

**Usage**

```
read_md(file)
```

**Arguments**

file	Filename to read from. Must be either .Rmd or .qmd file.
------	--

## Details

`read_md()` is unusual in that it calls `rmarkdown::render()` or `quarto::quarto_render()` internally to ensure code blocks and snippets are parsed correctly. This ensures dynamic content is rendered correctly in the resulting EML document, but makes this function considerably slower than a standard import function. Conceptually, therefore, it is closer to a renderer with output type tibble than a traditional `read_` function.

This approach has one unusual consequence; it prevents 'round-tripping' of embedded code. That is, dynamic content in code snippets within the metadata statement is rendered to plain text in EML. If that EML document is later re-imported to Rmd using `read_eml()` and `write_md()`, formerly dynamic content will be shown as plain text.

Internally, `read_md()` calls `lightparser::split_to_tbl()`.

## Value

`read_md()` returns an object of class `tbl_df`, `tbl` and `data.frame` (i.e. a tibble).

## Examples

```
source_file <- system.file("extdata",
                           "bionet_metadata.Rmd",
                           package = "delma")

read_md(source_file)
```

---

`use_metadata_template` Write an example metadata statement to disk

---

## Description

This function places a metadata template at the address specified by "file", defaulting to "metadata.Rmd" in the working directory. The template is built in such a way that standard rendering with `rmarkdown` or `Quarto` to HTML or PDF will function; but also that it renders to valid EML when processed using `read_md()` and `write_eml()`.

## Usage

```
use_metadata_template(file = NULL, overwrite = FALSE, quiet = FALSE)
```

## Arguments

<code>file</code>	(string) A name for the resulting file, with either <code>.Rmd</code> or <code>.qmd</code> as a suffix. If <code>NULL</code> will default to <code>metadata.md</code> .
<code>overwrite</code>	(logical) Should any existing file be overwritten? Defaults to <code>FALSE</code> .
<code>quiet</code>	(logical) Should messages be suppressed? Defaults to <code>FALSE</code> .



**Value**

Doesn't return anything to the workspace; called for the side-effect of placing a metadata statement in the working directory.

**Examples**

```
use_metadata_template("example.Rmd")
```

---

write_eml	<i>Write an EML-formatted metadata document</i>
-----------	---

---

**Description**

write\_eml() writes a tibble, list or xml\_document to an EML file. Note that EML files always have the file extension .xml.

**Usage**

```
write_eml(x, file)
```

**Arguments**

x	Object of any class handled by delma; i.e. tbl_df, list or xml_document.
file	Filename to write to

**Value**

Doesn't return anything; called for the side-effect of writing the specified EML file to disk.

**Examples**

```
source_file <- system.file("extdata",  
                           "bionet_metadata.Rmd",  
                           package = "delma")  
df <- read_md(source_file)  
write_eml(df, "example.xml")
```

---

`write_md`*Write a markdown-formatted metadata document*

---

**Description**

`write_md()` creates an Rmd or Qmd file from an EML file.

**Usage**

```
write_md(x, file)
```

**Arguments**

<code>x</code>	Object of any class handled by <code>delma</code> ; i.e. <code>tbl_lp</code> , <code>tbl_df</code> , <code>list</code> or <code>xml_document</code> .
<code>file</code>	Filename to write to. Must be either <code>.md</code> , <code>.Rmd</code> or <code>.qmd</code> file.

**Details**

Similar to [read\\_md\(\)](#), [write\\_md\(\)](#) is considerably less generic than most `write_` functions. If `x` is an `xml_document` this should convert seamlessly; but lists or tibbles that have been manually formatted require care. Internally, `write_md()` calls [lightparser::combine\\_tbl\\_to\\_file](#).

**Value**

Doesn't return anything; called for the side-effect of writing the specified markdown file to disk.

**Examples**

```
source_file <- system.file("extdata",  
                           "bionet_metadata.xml",  
                           package = "delma")  
df <- read_eml(source_file)  
write_md(df, "example.Rmd")
```

# Index

`as_eml_list`, [2](#)  
`as_eml_tibble`, [3](#)  
`as_eml_xml`, [4](#)  
`as_lp_tibble`, [5](#)  
  
`check_metadata`, [6](#)  
  
`lightparser::combine_tbl_to_file`, [10](#)  
`lightparser::split_to_tbl`(), [8](#)  
  
`quarto::quarto_render`(), [8](#)  
  
`read_eml`, [7](#)  
`read_eml`(), [8](#)  
`read_md`, [7](#)  
`read_md`(), [8](#), [10](#)  
`rmarkdown::render`(), [8](#)  
  
`use_metadata_template`, [8](#)  
  
`write_eml`, [9](#)  
`write_eml`(), [8](#)  
`write_md`, [10](#)  
`write_md`(), [8](#), [10](#)  
  
`xml2::read_xml`, [6](#)